IN THE SPECIFICATION

Please replace the paragraph at page 1, line 2, starting with "This application is . . ." as follows:

This application is a continuation of United States Application Serial No. 09/909,701 filed July 20, 2001, now U.S. Patent No. 6,678,798 issued January 13, 2004, which claims the benefit of U.S. Provisional Application Serial No. 60/219,953 filed July 20, 2000.

Please replace the paragraph beginning on page 12, line 1, and starting with "Information in the sharing vector . . ." as follows:

Information in the sharing vector tracks the location of exclusive or shared copies of a cache line as required to enforce the protocol that maintains coherence between those copies and the home location of the cache line. The sharing vector may be used in one of three ways depending on the directory state. The sharing vector may be in a pointer format as a binary node pointer to a single processor node or input/output node. This format is used when the state is EXCL as well as in most transient states. The sharing vector may be in a pointer timer format as a combination of an input/output read timer and a binary node pointer. This format handles the read exclusive read-only (RDXRO) transaction. The sharing vector may be in a bit vector format as a bit vector of sharers. The field is preferably partitioned into a plane bit vector, a row bit vector, and a column bit vector. This format is used when the cache line is in a SHRD state. Examples of the use of the sharing vector can be found in copending U.S. Application Serial No. 08/971,184 entitled "Multiprocessor Computer System and Method for Maintaining Cache Coherence Utilizing a Multi-dimensional Cache Coherence Directory Structure", now U.S. Patent No. 6,633,958 and in copending U.S. Application Serial No. 09/910,630 entitled "Method and System for Efficient Use of a Multi-dimensional Sharing Vector in a Computer System", both of which are incorporated herein by reference.

Please replace the paragraph beginning on page 42, line 22, and starting with "FIGURE 3 also shows . . ." as follows:

FIGURE 3 also shows the events that occur when a remote processor seeks access to the cache line prior to processing of the implicit writeback. After processor 100 initiates an implicit writeback to front side bus processor interface 24, a remote processor 200 initiates a read request to memory directory interface unit 22. Memory directory interface unit 22 initiates an intervention for transfer to front side bus processor interface 24 since it thinks that processor 100 is the current owner of the cache line. Memory directory interface unit 22 will also send a speculative response to remote processor 200 since it thinks it has the latest copy of the cache line. Front side bus processor interface 24 receives the intervention but knows it has an implicit writeback to process. The intervention is placed on hold and the implicit writeback is sent to memory directory interface unit 22. Upon processing the implicit writeback, memory directory interface unit 22 sends the writeback ACK. Front side bus processor ~~interface 22~~ interface 24 receives the writeback ACK and determines if there is a pending writeback in its queue 102. If so, front side bus processor interface 24 sends out the pending writeback to memory directory interface ~~unit 24~~ unit 22 and also sends out a response to remote processor 200 since it has the latest copy of the cache line. In this manner, the latest copy of the cache line may be provided for read requests while a writeback is pending.